LA-UR-13-26190
Approved for public release; distribution is unlimited.

Title: Numerically Intensive and Data Intensive Computing: Issues, Approaches and Leveraging

Author(s): Ahrens, James P.
Sewell, Christopher Meyer

Intended for: Distribute to attendees of materials science SRG Workshop

Issued: 2013-08-05

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer,is operated by the Los Alamos National Security, LLC for the National NuclearSecurity Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Departmentof Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

U.S. DEPARTMENT OF ENERGY | Office of Science | Advanced Simulation & Computing™

Numerically Intensive and Data Intensive Computing: Issues, Approaches and Leveraging

James Ahrens and Chris Sewell

Los Alamos National Laboratory

Chris Mitchell, Turab Lookman, Ollie Lo

Scope of this talk

- **Discuss the impact of the technologies developed for numerically intensive/ exascale computing on data-intensive computing and the broader industrial computing infrastructure.**

- **Framing the problem**
  - Numerically Intensive
    - Exascale Architectural Issues
  - Data Intensive
    - Characteristics
- **Probable impacts of the pursuit of exascale on specific technical areas and industry**
- **Conclusions**

Exascale Architectural Issues

- **Scaling standard solutions will not work**
  - Massive number of cores/data sizes magnify:
    - Power inefficiencies
    - The need to exploit concurrency for high-performance
    - Bandwidth needs
- **To achieve the next level of supercomputing performance we need to address these issues**
  - Solutions to these issues will impact data intensive computing industry

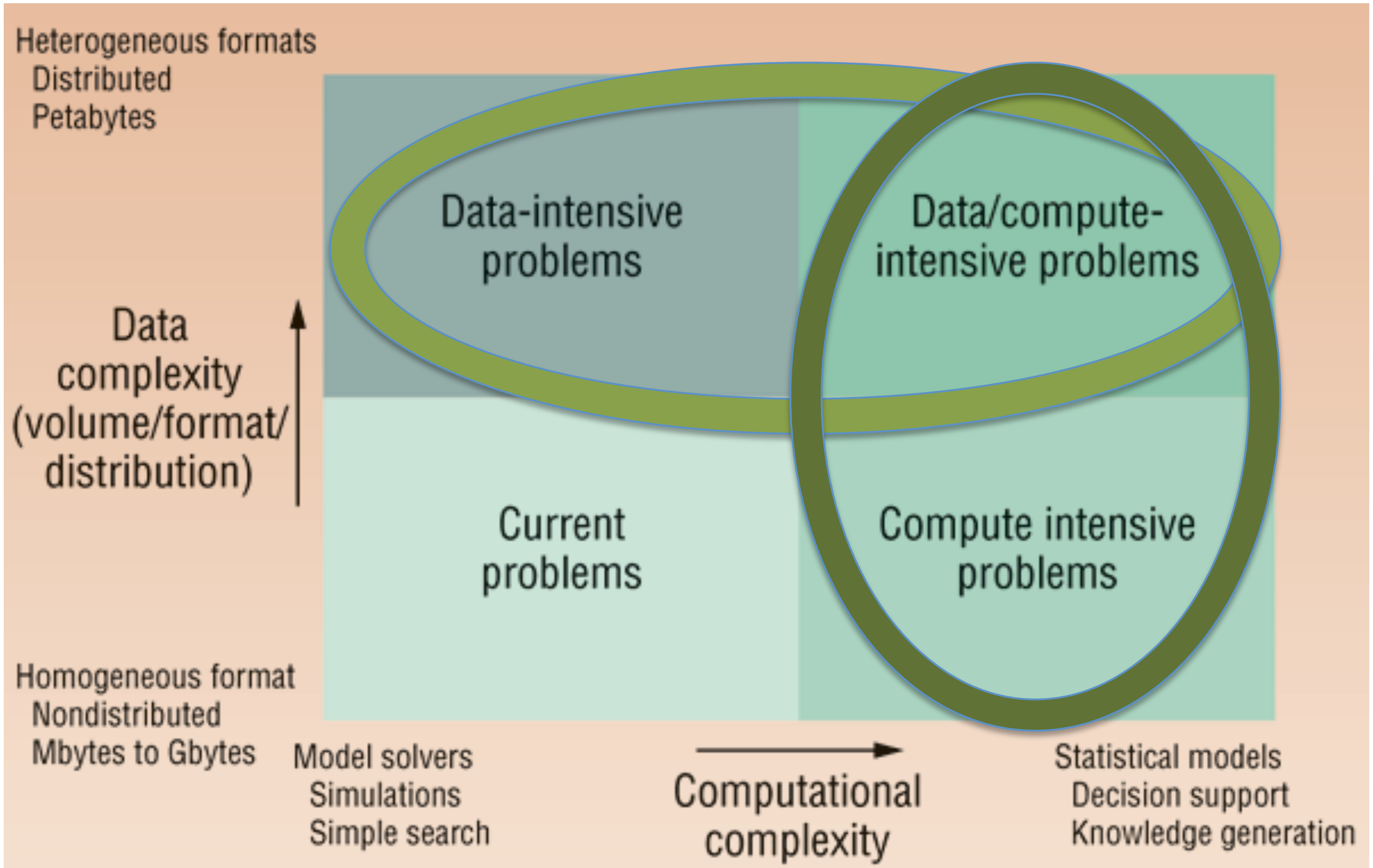
Data Intensive Approaches

- **This talk will focus on the most scalable data intensive approach:**
  - Map reduce ecosystem
    - Server infrastructure
      - Clusters of "a few thousand processors," with disk storage associated with each processor from pool of 10^6 processors
    - Large database community driver
      - Success of this approach - Thousands of processors, terabytes of data, tenth of second response time
- **Other data intensive approaches include:**
  - No-SQL datastores
  - Massive graph processing
  - Time critical financial analytics

A characterization of numerically/compute intensive versus data intensive approaches

		Numerically Intensive	Data Intensive
Hardware	Nodes and Interconnect	High performance and power	Lower performance and power
	Storage	Separate, independent	Integrated
SW	Synchronization	Tightly coupled	Loosely coupled
	Reliability	Checkpoint restart	Replication
Workload	Number of Users	Single per node	Multiple per node
	Data	Dynamic, heterogeneous (unstructured grid)	Static, homogeneous (text, images)
	Algorithms	Global	Distributed
Workflow	Scheduling	Batch	Interactive
	Resource Tasking	Specific	Abstract
	Analysis	Offline post-processing	Online
	I/O	Bulk parallel writes	Streaming writes

Numerically/Compute Intensive and Data Intensive – Intersecting Approaches



Ian Gorton, Paul Greenfield, Alex Szalay, Roy Williams, "Data-Intensive Computing in the 21st Century," Computer, pp. 30-32, April, 2008

Leveraging the Success of Both Approaches

- **Commoditization of scalable data intensive approaches**
  - Success for high-performance computing community
- **Data intensive workloads currently simpler than numerically intensive**
- **Competitive pressure to improve data intensive algorithms and services**
  - 60% potential increasing retailers' operating margins possible with big data
  - "Big data: The next frontier for innovation, competition, and productivity", McKinsey Global Institute, May 2011

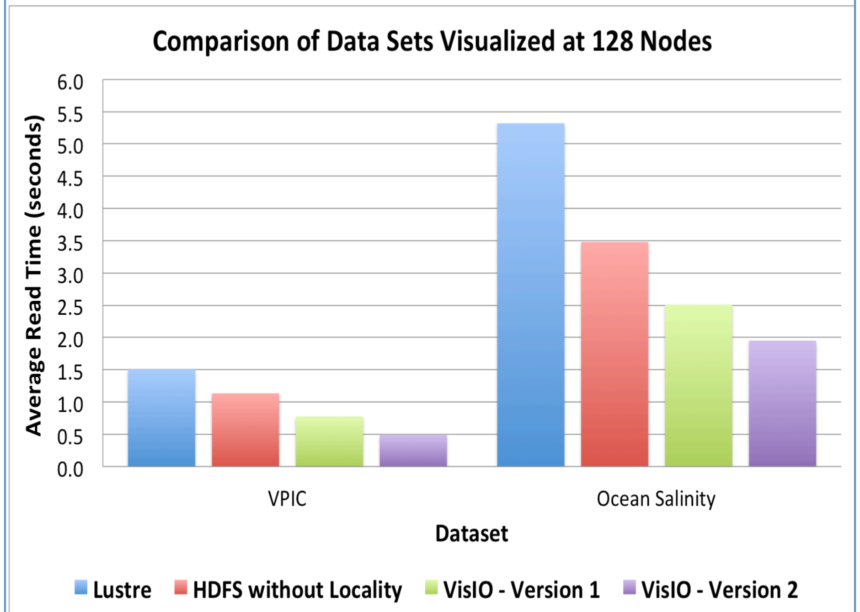
Opportunity for cross pollination…

Is it possible to?

- Simplify the numerically intensive approach and still achieve high performance?
- Increase the sophistication of data intensive approach and while retaining simplicity and flexibility?

An Example of Integrating Numerically and Data Intensive Approaches / VISIO

- **Use Hadoop Distributed File System (HDFS) instead of Lustre**
  - with ParaView visualization application
  - *3x improvement and reduced variance* in read times
- **Compose relevant parts of each ecosystem**
  - Did not use map reduce scheduler

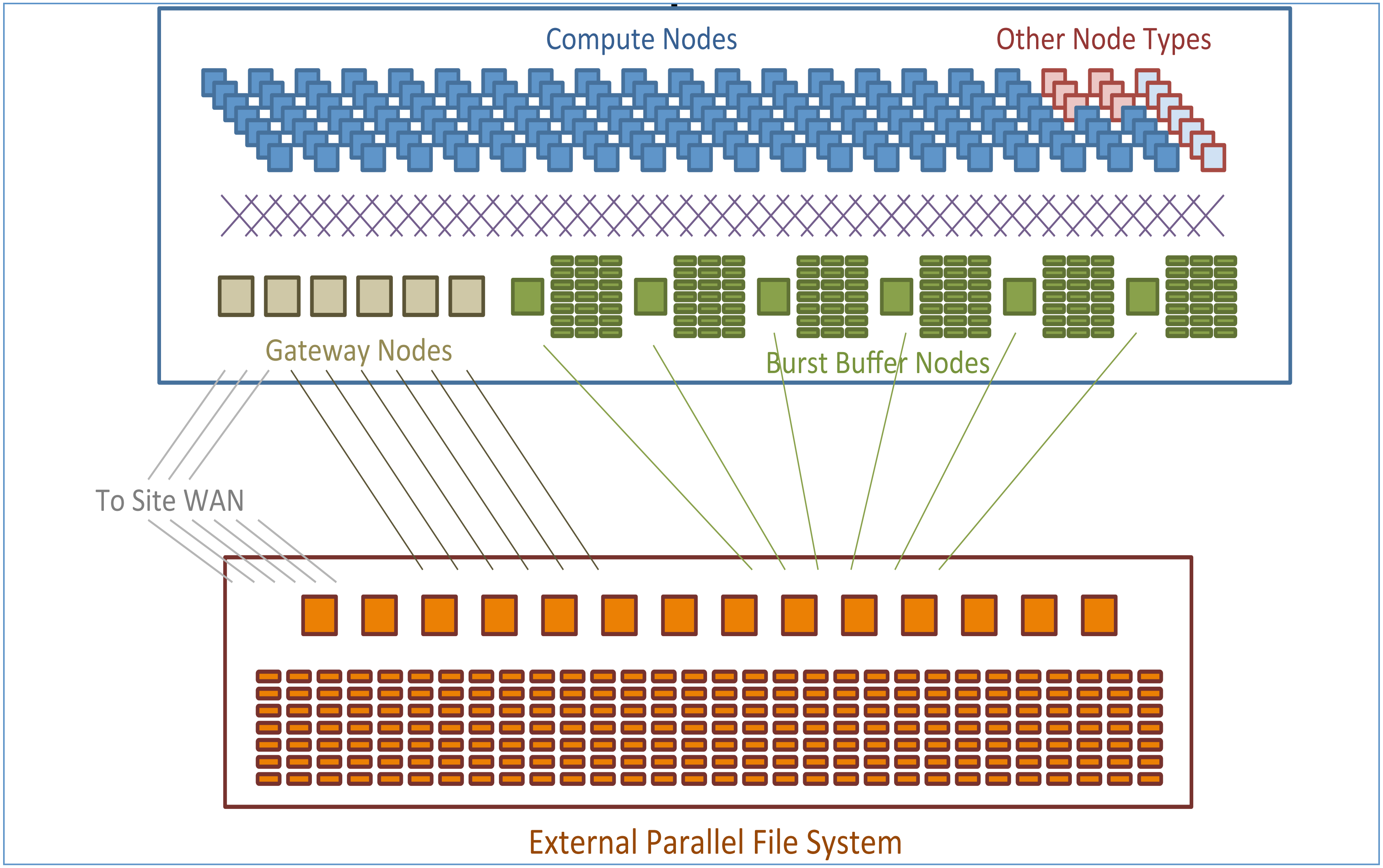


C. Mitchell, J. Ahrens, and J. Wang. "VisIO: Enabling Interactive Visualization of Ultra-Scale, Time Series Data via High-Bandwidth Distributed I/O Systems". IEEE International Parallel and Distributed Processing Symposium, May 2011.

A Materials Example

- **Target hardware – exascale architecture with burst buffer**
- **Application - CoGL**
- **Software solution - PISTON**

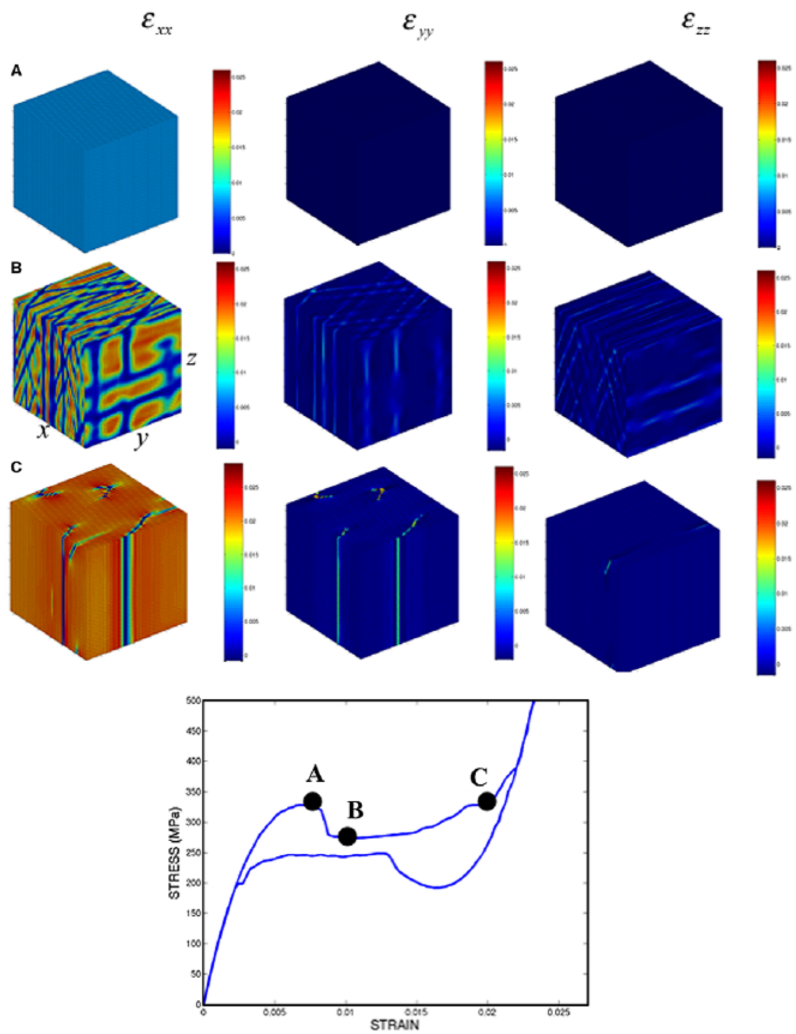
Potential Exascale Architecture



Burst Buffer Overview

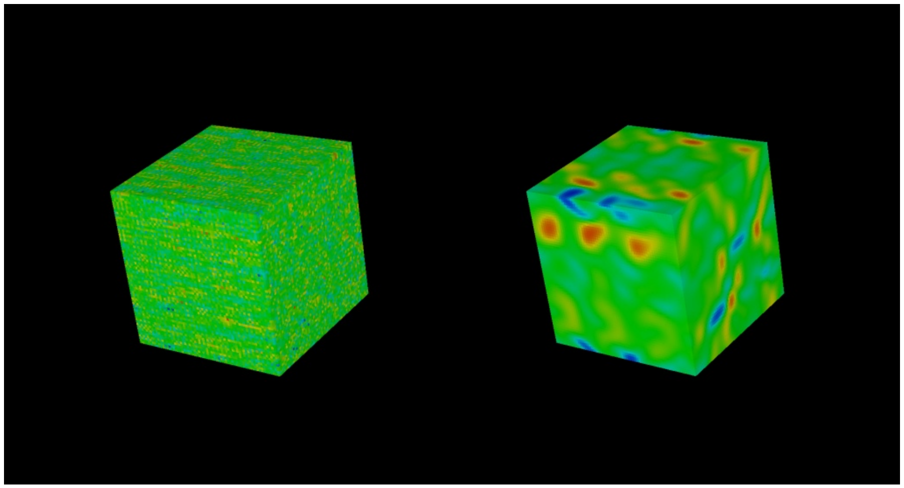
- **A Burst Buffer is a device designed to shield compute nodes from the bandwidth limits of the disk-based parallel file system by providing a pool of fast flash memory.**
- **Current Prototypes:**
  – A set of x86_64 servers with locally attached disks that are attached to both the compute fabric as well as the storage fabric.
- **Primary Use: Faster Checkpoint/Restart**
- **Secondary Use: Perform In-Transit Data Analysis**
  – Focus of funded LDRD-ER exploration.

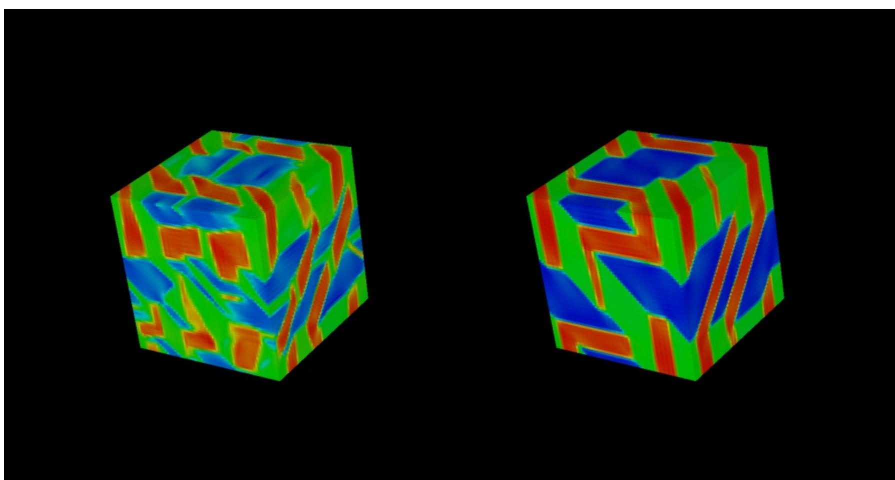
Application - CoGL



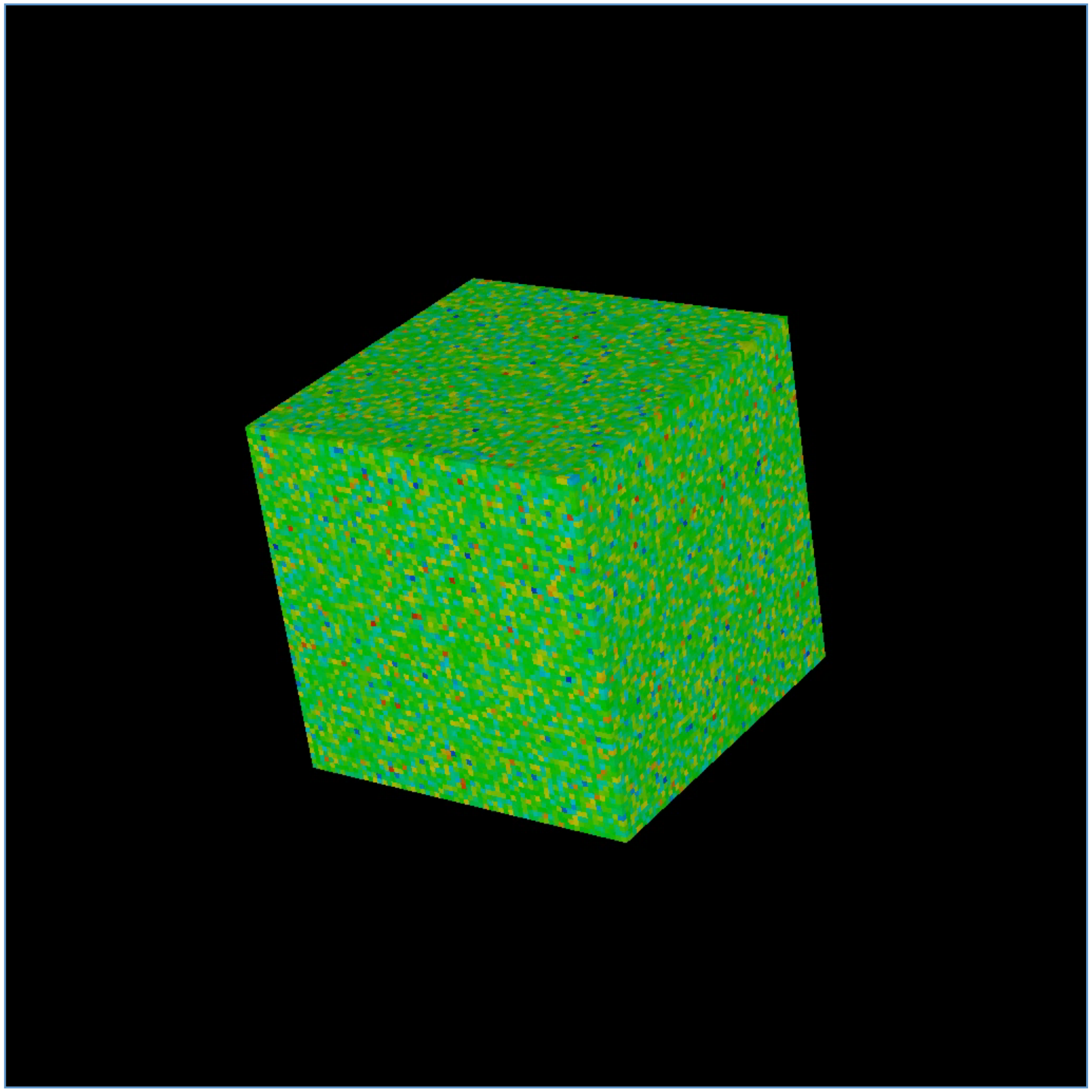
- A proxy app being developed for the Exascale Co-Design Center for Materials in Extreme Environments
- Stand-alone meso-scale simulation code
- Studies pattern formation in ferroelastic materials using the Ginzburg–Landau approach
- Models cubic-to-tetragonal transitions under dynamic strain loading
- Based on a nonlinear elastic free-energy in terms of the appropriate strain fields

Portable, Parallel CoGL with in-situ



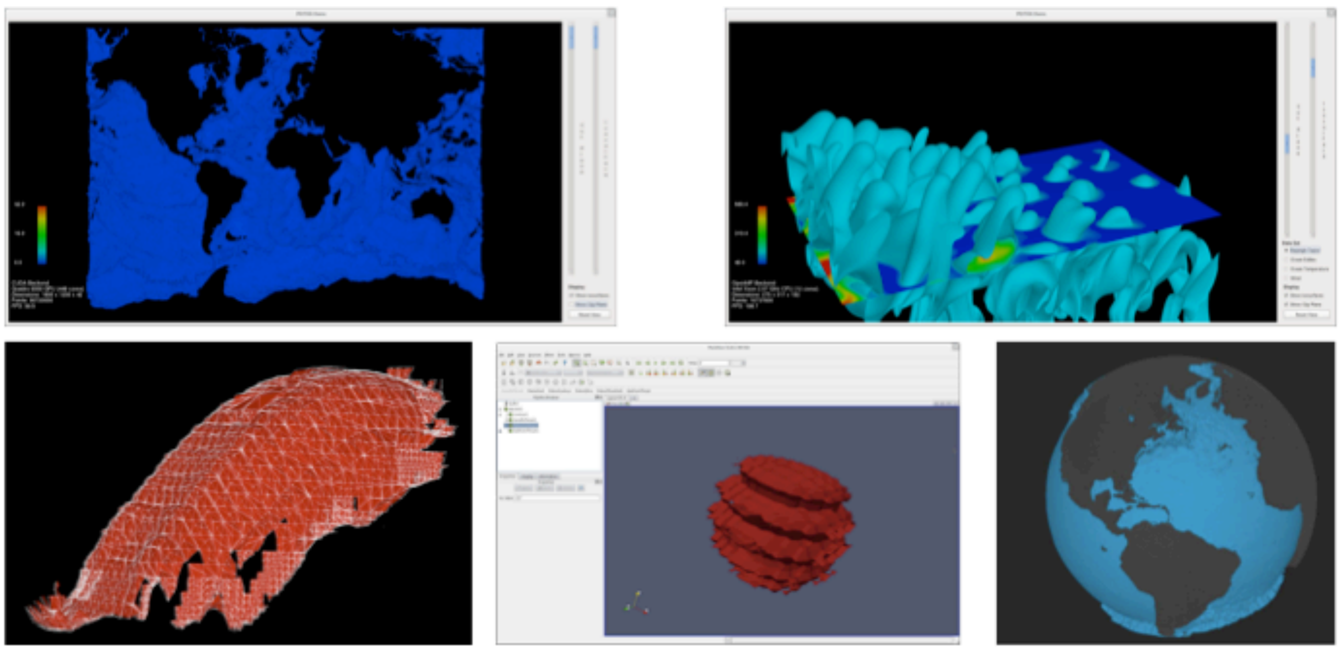


- Simulation code and in-situ viz implemented using PISTON, our portable, data-parallel viz and analysis library built on NVIDIA's Thrust library
- Allows the exact same code to run efficiently on all parallel architectures supported by backend (currently including GPUs with CUDA and multi-core CPUs with OpenMP)
- When running on GPUs, "interop" allows fast rendering by eliminating unnecessary data transfers
- Much faster than original Fortran code



PISTON: A Portable Data-Parallel Visualization and Analysis Framework

- Goal: Portability and performance for visualization and analysis operators on current and next-generation supercomputers
- Main idea: Write operators using only data-parallel primitives (scan, reduce, etc.)
- Requires architecture-specific optimizations for only for the small set of primitives
- PISTON is built on top of NVIDIA's Thrust Library



Motivation and Background

- Current production visualization software does not take full advantage of acceleration hardware and/or multi-core architecture
- Research on accelerating visualization operations are mostly hardware-specific; few were integrated in visualization software
- Standards such as OpenCL may allow program to run cross-platform, but usually still requires many architecture specific optimizations to run well
- Data parallelism: independent processors performs the same task on different pieces of data (see Blelloch, "Vector Models for Data Parallel Computing")
- Due to the massive data sizes we expect to be simulating we expect data parallelism to be a good way to exploit parallelism on current and next generation architectures
- Thrust is a NVidia C++ template library for CUDA. It can also target other backends such as OpenMP, and allows you to program using an interface similar the C++ Standard Template Library (STL)

Brief Introduction to Data-Parallel Programming and Thrust

What algorithms does Thrust provide?

- Sorts
- Transforms
- Reductions
- Scans
- Binary searches
- Stream compactions
- Scatters / gathers

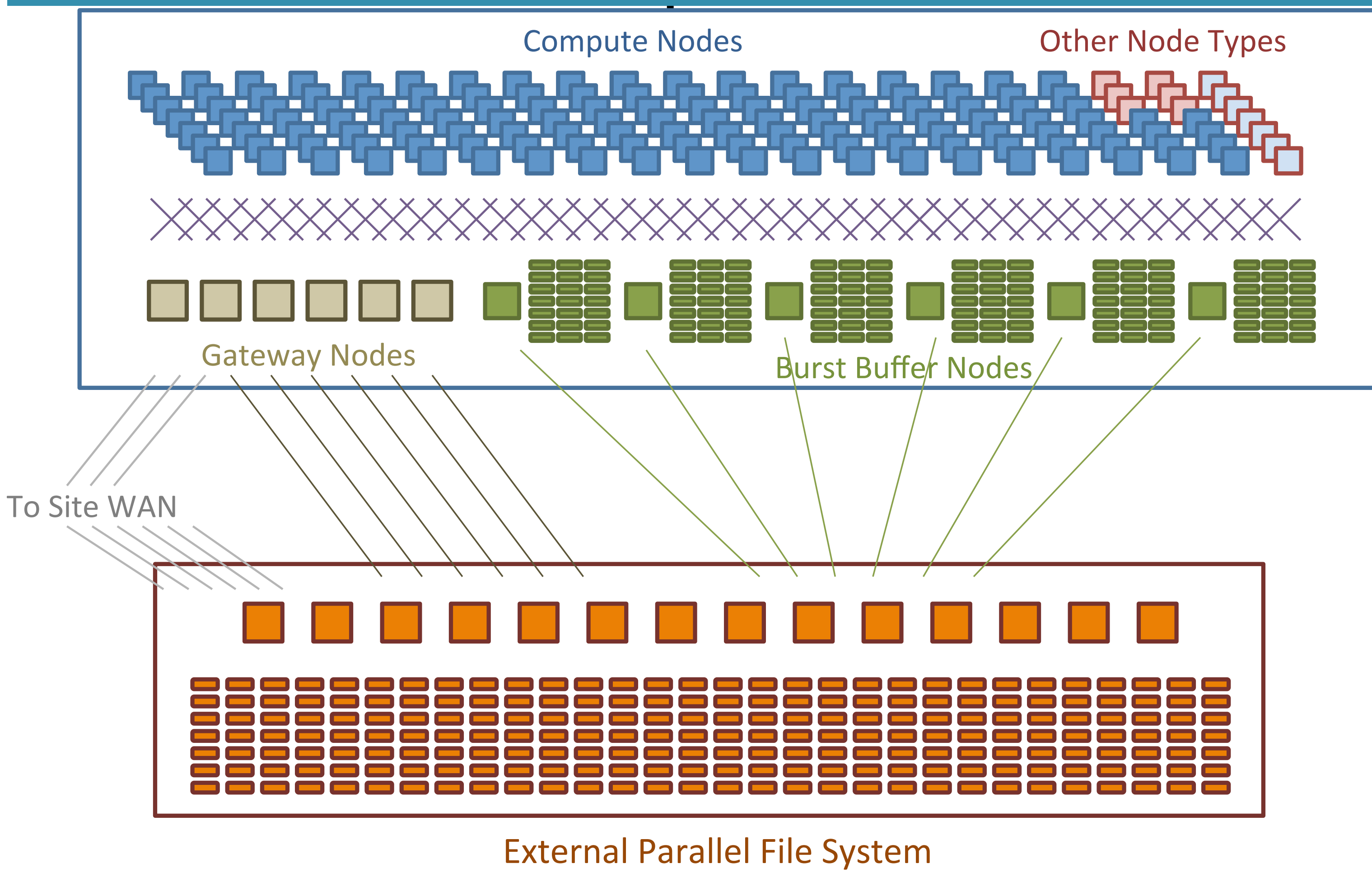
```
input                       4  5  2  1  3
-----------------------------------------
transform(+1)               5  6  3  2  4
inclusive_scan(+)           4  9 11 12 15
exclusive_scan(+)           0  4  9 11 12
exclusive_scan(max)         0  4  5  5  5
transform_inscan(*2,+)      8 18 22 24 30
for_each(-1)                3  4  1  0  2
sort                        1  2  3  4  5
copy_if(n % 2 == 1)         5  1  3
reduce(+)                             15

input1                      0  0  2  4  8
input2                      3  4  1  0  2
-----------------------------------------
upper_bound                 3  4  2  2  3
permutation_iterator        4  8  0  0  2
```

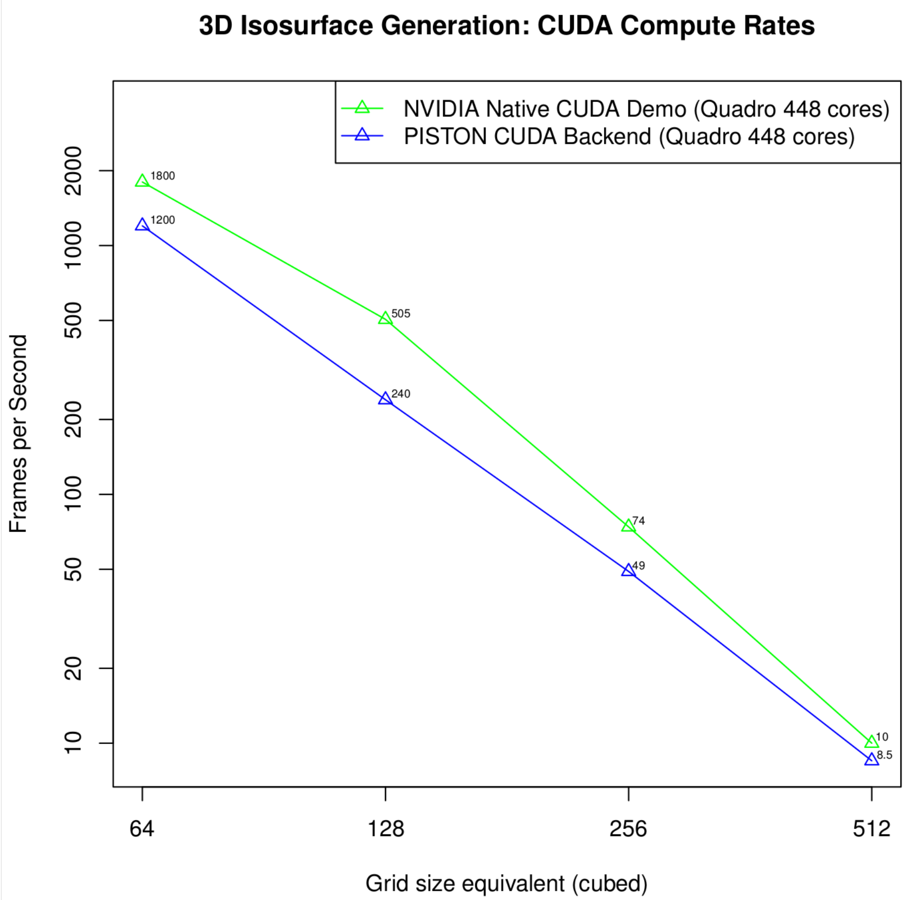
Challenge: Write operators in terms of these primitives only

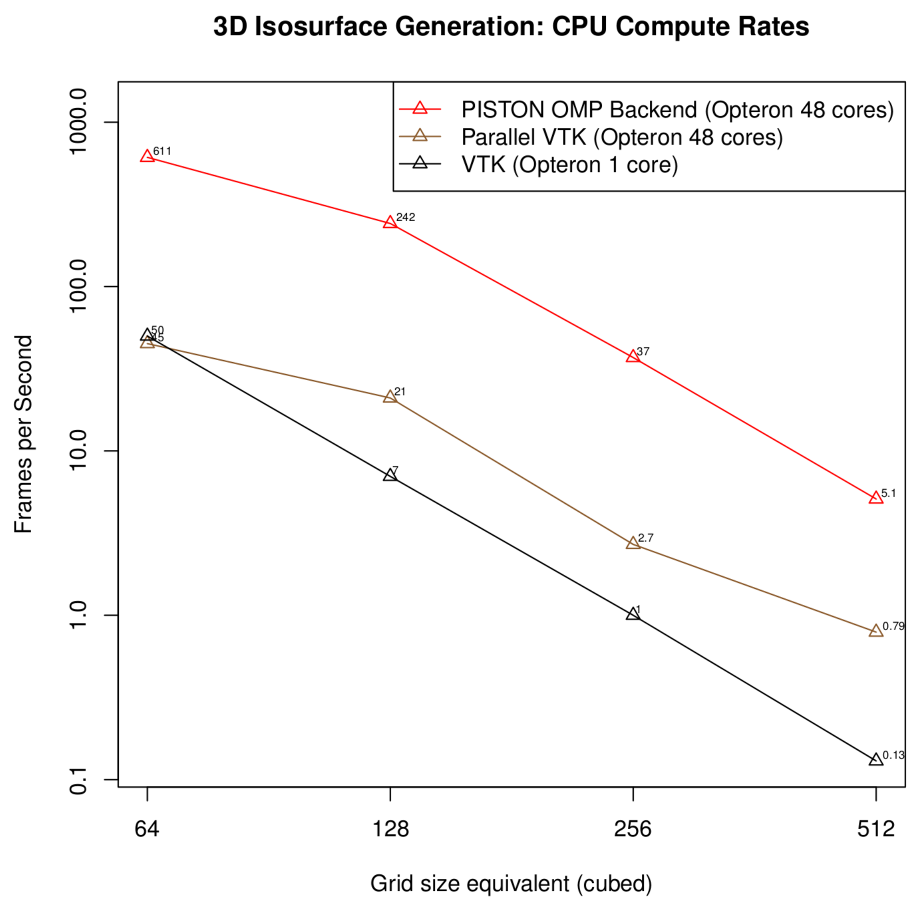
Reward: Efficient, portable code

Potential Exascale Architecture

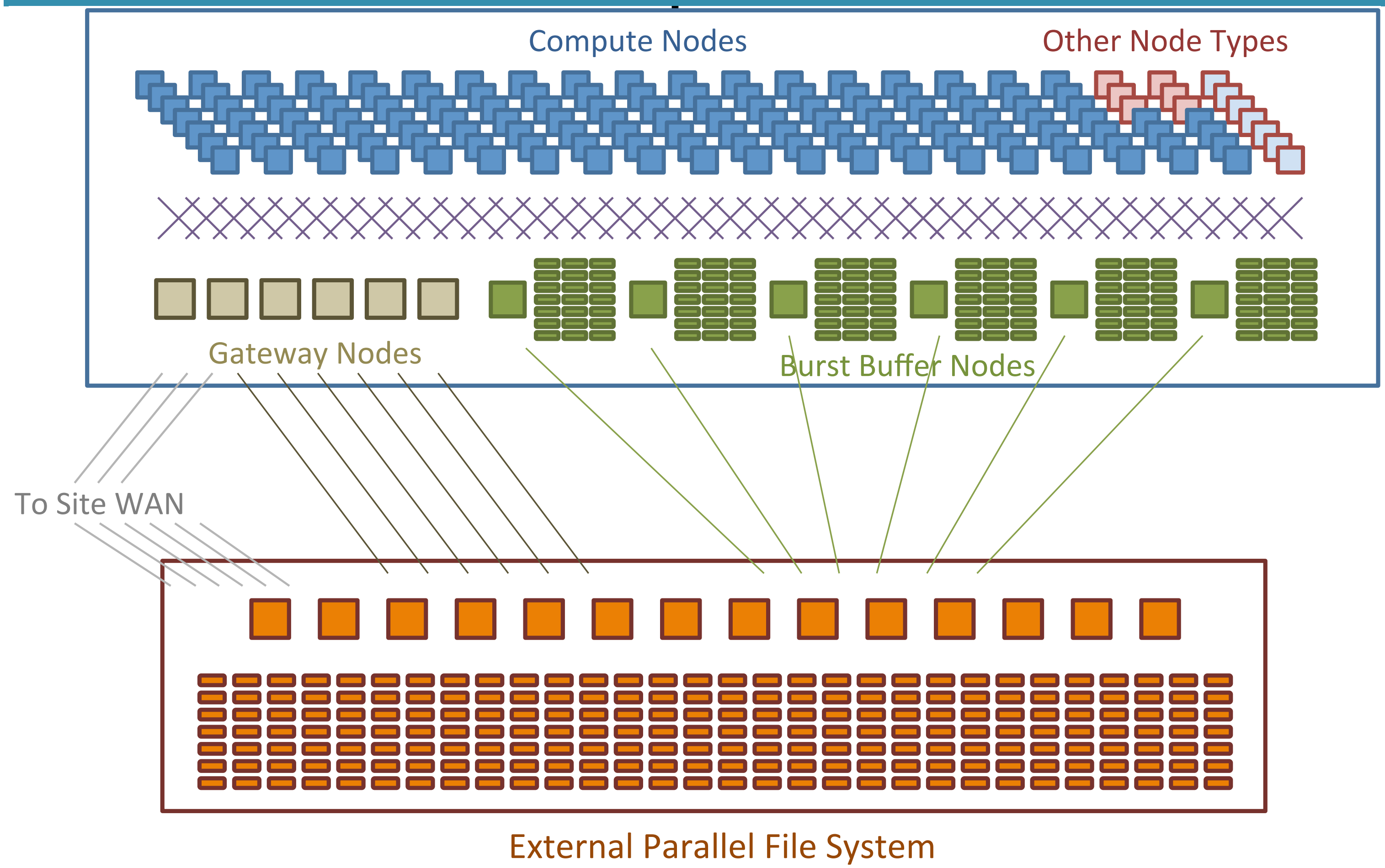


PISTON: Single Node Accelerated Architectures



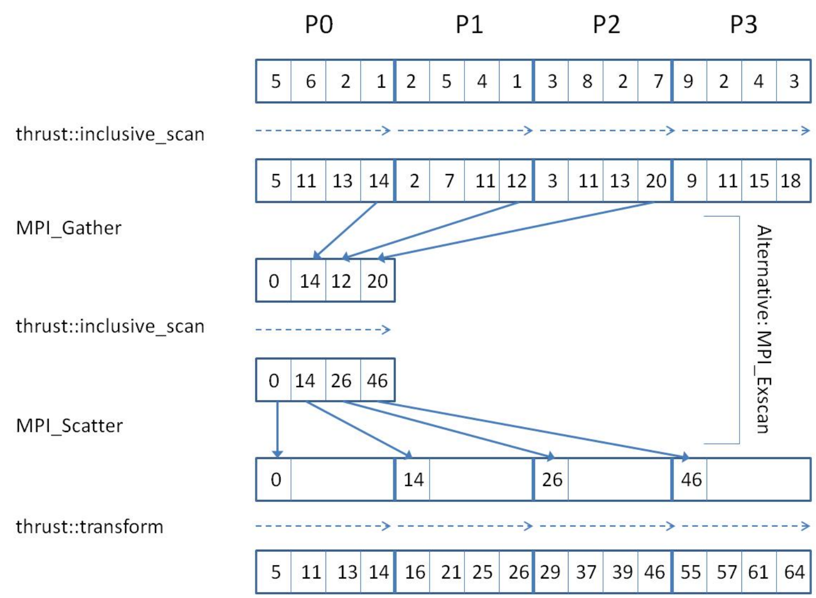


Potential Exascale Architecture

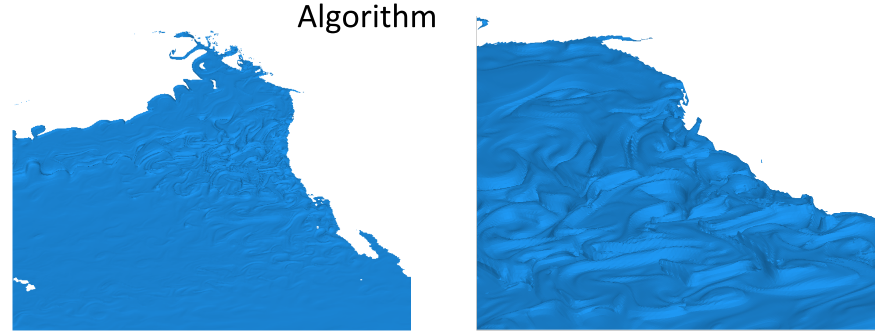


PISTON: Distributed Memory Architectures

- Inter-node (distributed memory) parallelism
  - VTK Integration handles domain decomposition / image compositing
  - Distributed implementations of Thrust primitives using MPI
    - User can treat data as single vectors even though values are distributed across nodes
    - Regular Thrust primitives are called for on-node work, so it takes advantage of parallelism both on nodes and across nodes
    - Implemented isosurface and KD-tree construction algorithms using distributed PISTON

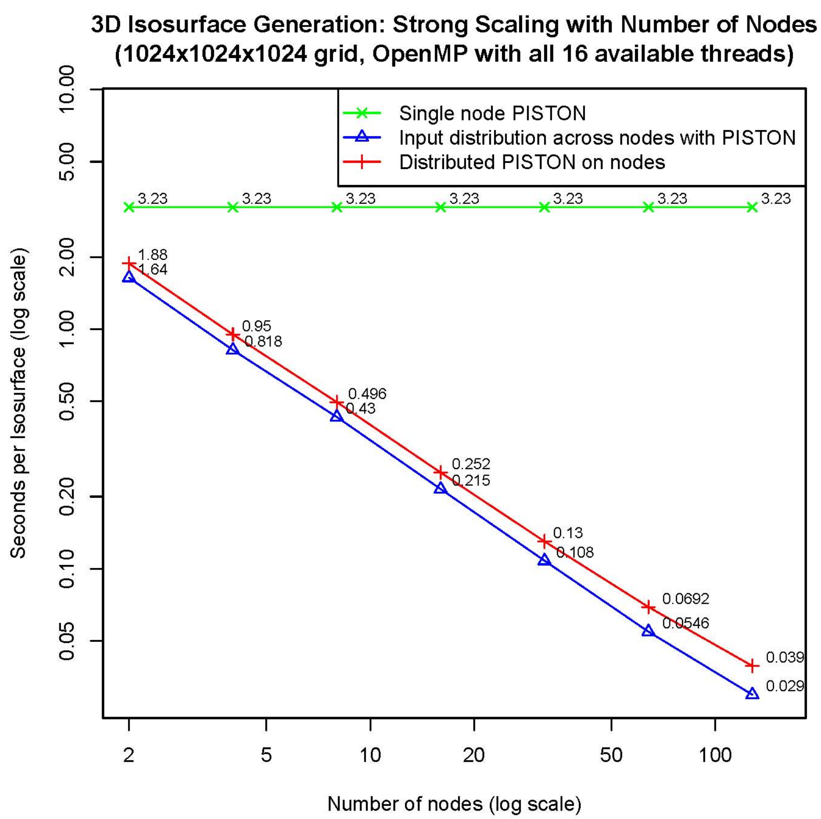


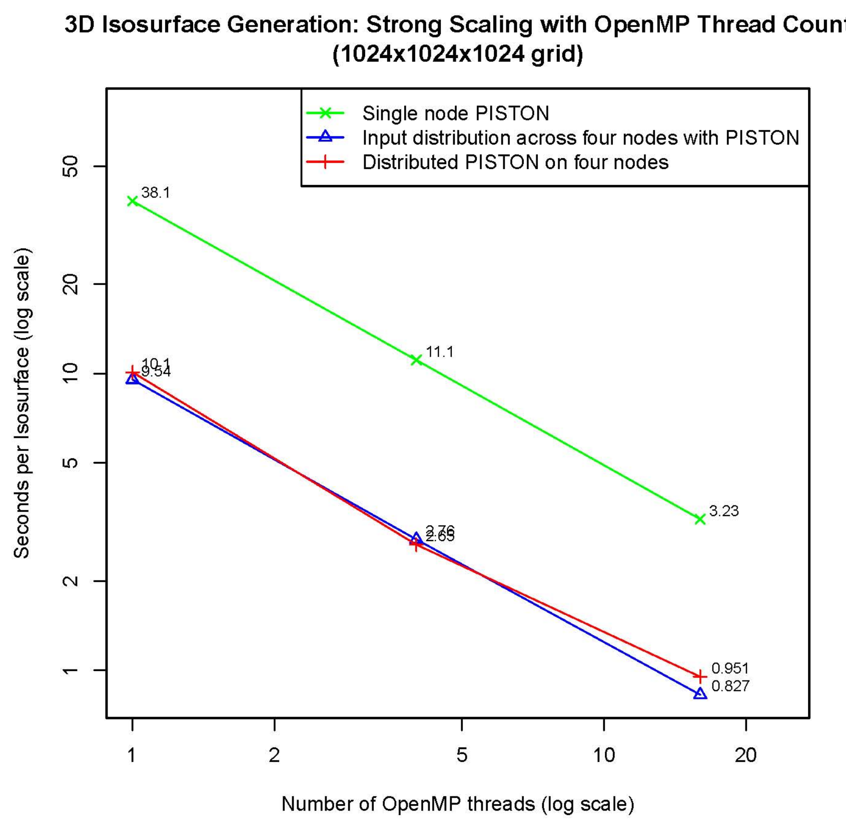
Distributed Scan Algorithm



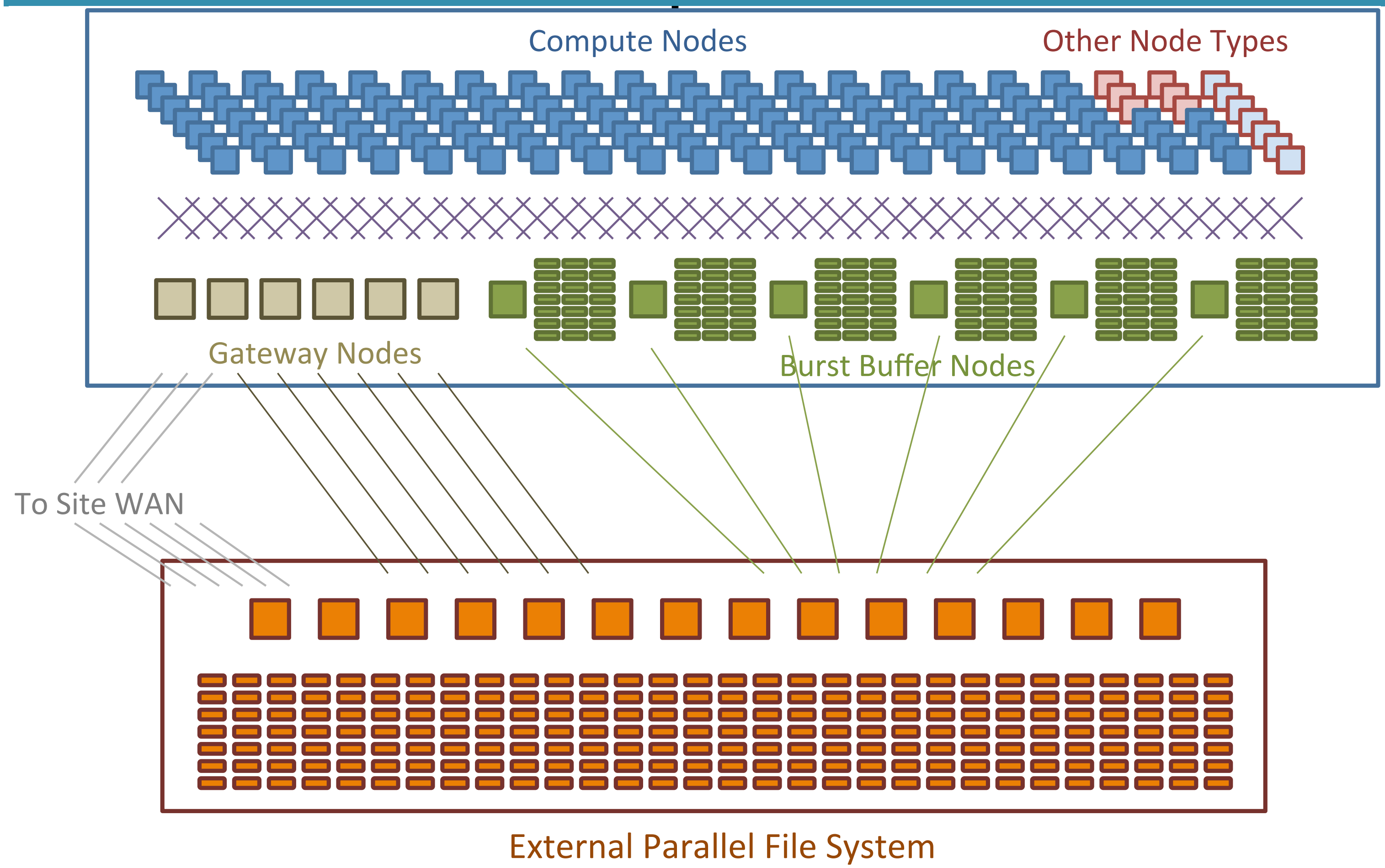
Isosurface of 3600x2400x42 ocean temperature data computed on 4 GPUs

PISTON: Distributed Memory Architectures





Potential Exascale Architecture



PISTON: Streaming & Data Architectures

- **Extend PISTON to handle streaming data**
  - Compute on data located anywhere without requiring a pre-load into node memory.
  - Data can be streamed from disk, compute nodes, external sources (including sensors), etc.
- **Add Data Architecture Support to PISTON:**
  - Execute PISTON functions / partial pipelines where data resides rather than moving the data.
    - Ex.) Data Reduction Operations
  - Ex.) Execute on burst buffer nodes while data is resident rather than having to reload from disk at a latter time.
  - Explore possibility on running on additional architectures including storage controllers (ARM), Power (IBM data solutions), etc.